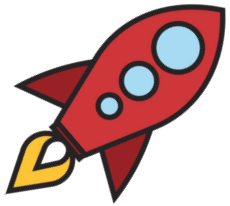


Linux in Space: Fault detection, recovery and fault-tolerant system designs

lenka.koskova.triskova@tul.cz, lukas.mazl@tul.cz, tomas.novotny@vzlu.cz

2024-09-16

Outline

- CubeSats and NewSpace
- Fault tolerant design principles
- How it relates to Linux
- VZLUSAT-2/ZHP
  - Hardware
  - U-Boot configuration
  - Linux configuration

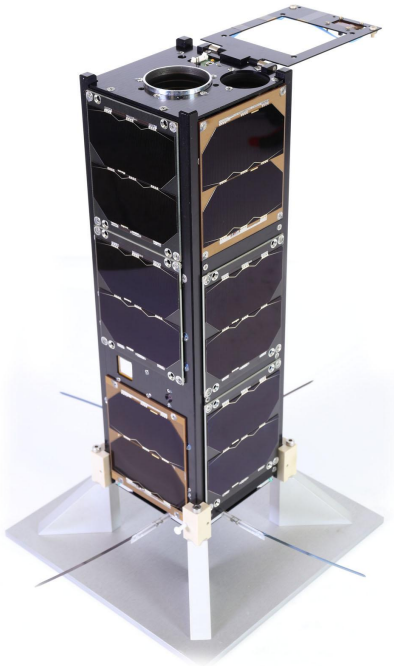
Takeaway

- What limits HW/SW design for space
- What to keep in mind when building a CubeSat?
- How to analyse my system?
- A real flight proven space use-case

Linux4Space Use-Case

- CubeSats.
  - Size: ~carton of milk
  - Mass: ~5 kg
  - Limited resources (e.g., power, budget)
  - Short lived missions
- Payload system, not the central platform (mission) system.
- No GUI.
- Space environment ready (reliable in radiation, high temperature)
- Designed to be as close to ESA Standards and recommendations as possible.
- Two Yocto layers (system and applications)

Newspace: Standard HW, if one CubeSat fails, build another one (it starts to be so cheap and available).



Linux4Space Demo Mission

- CubeSat demonstration for development.
- TULSat includes:
  - RPi 4 as onboard computer (OBC).
  - Beaglebone Black as payload computer.
- Communication – Cubesat Space protocol (UART interface)
- OBC sends telemetry data only via wifi/ethernet in first demo, finally the KISS/AX.25 is to be implemented

Why it is a bit more complicated in space?

- **Radiation** - electronic failures, fast shutdowns
- **Temperature issues -** extreme temperature variations and no cooling.
- **Heavy vibrations** during transport on a rocket system.
- **Remote system** slow links, low bandwidth, accessible only when visible (20 minutes communication window).
- **No physical access** to the device to diagnose the problem.
- **Costs** - you pay a lot to go to space and you have just one device, if it fails, you fail.

From the satellite designers:

- The biggest challenge is the radiation and limited cooling.
- The biggest design constraint is the limited power consumption.

Fault tolerant system design

Anything that can go wrong will go wrong (Murphy's law).

Fault tolerance – the system's ability to overcome and handle failures.

Fault source: SW, physical environment, common environment (e. g. system overload).

FDIR - Fault Detection, Isolation and Recovery – “detect the error and get system working again” – watch points & action points

In CubeSats, SoCs are the most frequent source of system failure (https://liacs.leidenuniv.nl/~plaata1/theses/ChristianFuchs.pdf)

FMEA/FMECA - since Apollo

- Failure Mode Effects (Criticality) Analysis
- Possible failures classified at selected levels of severity, probability of failure estimated – the output is criticality Matrix

Probability Severity	1	2	3	4	5
1	Low	Low	Low	Low	Moderate
2	Low	Low	Low	Moderate	High
3	Low	Low	Moderate	Moderate	High
4	Low	Moderate	Moderate	High	Unacceptable
5	Moderate	Moderate	High	Unacceptable	Unacceptable

Redundant designs

Spatial redundancy: Redundant systems working in parallel, output compared and selected (simple parts, ICs, PCs).

Dual redundancy (hot, cold), triple redundancy with voter

Cubesats are not large scale spacecrafts: KISS referenced in a lot in design descriptions.

43% of CubeSats do not use any redundancy due to budget, time or space constraints – https://doi.org/10.1016/j.micpro.2021.104312

Redundant design example

Aalto-1:

- 2 CPU, each 8 MB data flash, 8 MB NOR, 32 MB RAM, 256 MB NAND flash
- NAND flash for filesystem, each contains 2 images
  - 4 FS in total, ECC kernel driver protects data in the NAND flash
- NOR and data flash contain both the bootloader and the kernel (4 bootlader&kernel pairs in total)
- FDIR: hardware watchdog in CPU supported by kernel, userspace watchdog, communication watchdog
- ECC for memories and communication protocols

(Developing a Linux-Based Nanosatellite On-Board Computer: Flight Results from the Aalto-1 Mission, https://ieeexplore.ieee.org/document/8637100)

System (re)boot

- Re-booting can be frequent due to radiation shutdowns
- Redundant storage – Image selection on boot
- Flight proven “sophisticated way”: checksum of kernel/fs images.
- Flight proven “lets try way”: Primary and secondary SD card; try to boot the primary (3 times) if it fails, let's try the second one 3 times.

System updates

- OVA (“over the vacuum” update) ;)
- Due to slow communication and low bandwidth tricky and risky
- Checksums, and recovery operation must be ready

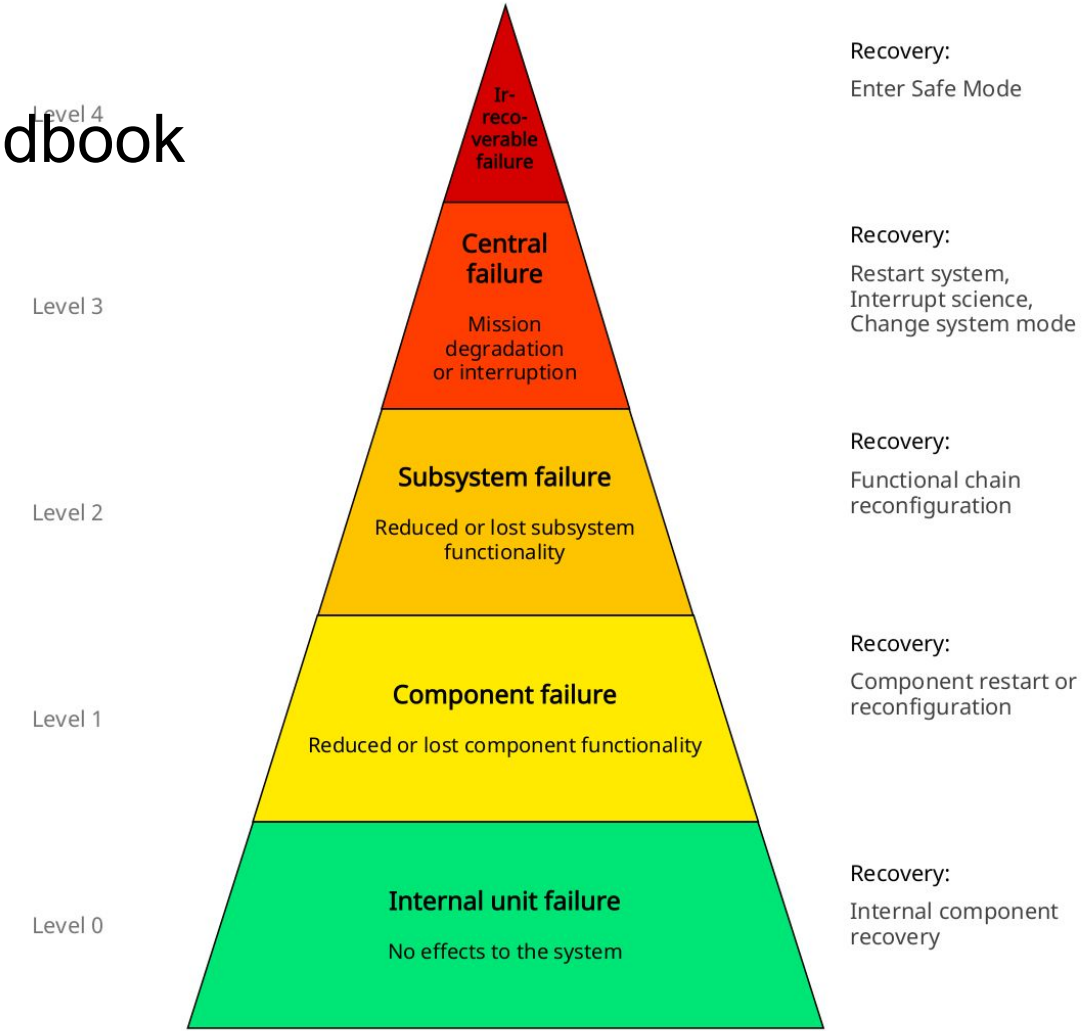
- Binaries update – the most frequent “mainstream”
- Partial binary update
- Compilation on-board

ESA/ECSS

- Faults shall be identified through analysis process: Fault Tree Analysis, FMEA/FMECA Analysis, Hazard Analysis, HW/SW Interaction Analysis
- ECSS-Q-ST-30-02C FMEA/FMECA, ECSS-Q-ST-40-09C Availability, ECSS-Q-ST-40-09C Hazard Analysis, ECSS-Q-ST-40-12C FTA, ECSS-Q-ST-30C Dependability (FMECA/FMEA/FTA), ECSS-Q-ST-80C SW Product Assurance (SW Criticality)
- **For CubeSats mostly too complex (in design references the authors mostly keep the KISS principle)**
- If you really want ECSS – SAVOIR FDIR Handbook from ESA – best practices and recommendations

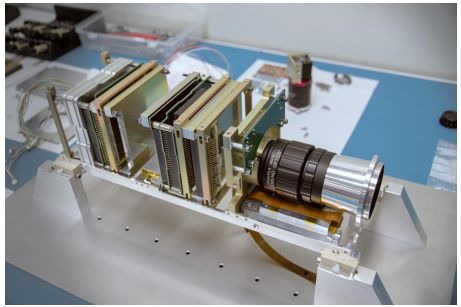
SAVOIR FDIR ESA Handbook

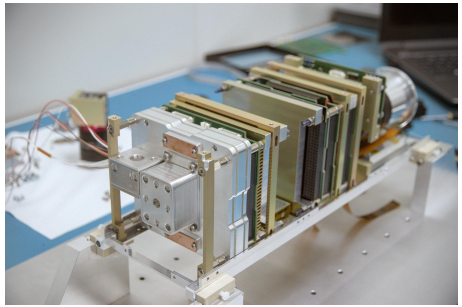
1. System availability and reliability **requirements** analysis/definition
2. **Concept** definition (redundancy, cross-strapping, triple-modular redundancy, high cost reliable components)
3. High level **architecture** design
4. **FMEA analysis** – list of all possible failure modes
5. **Implementation**

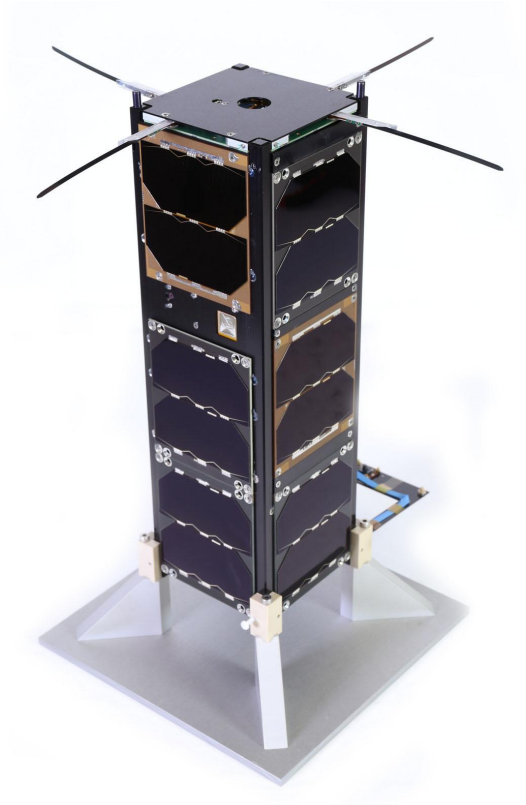


VZLÚ's approach on NewSpace 1/2

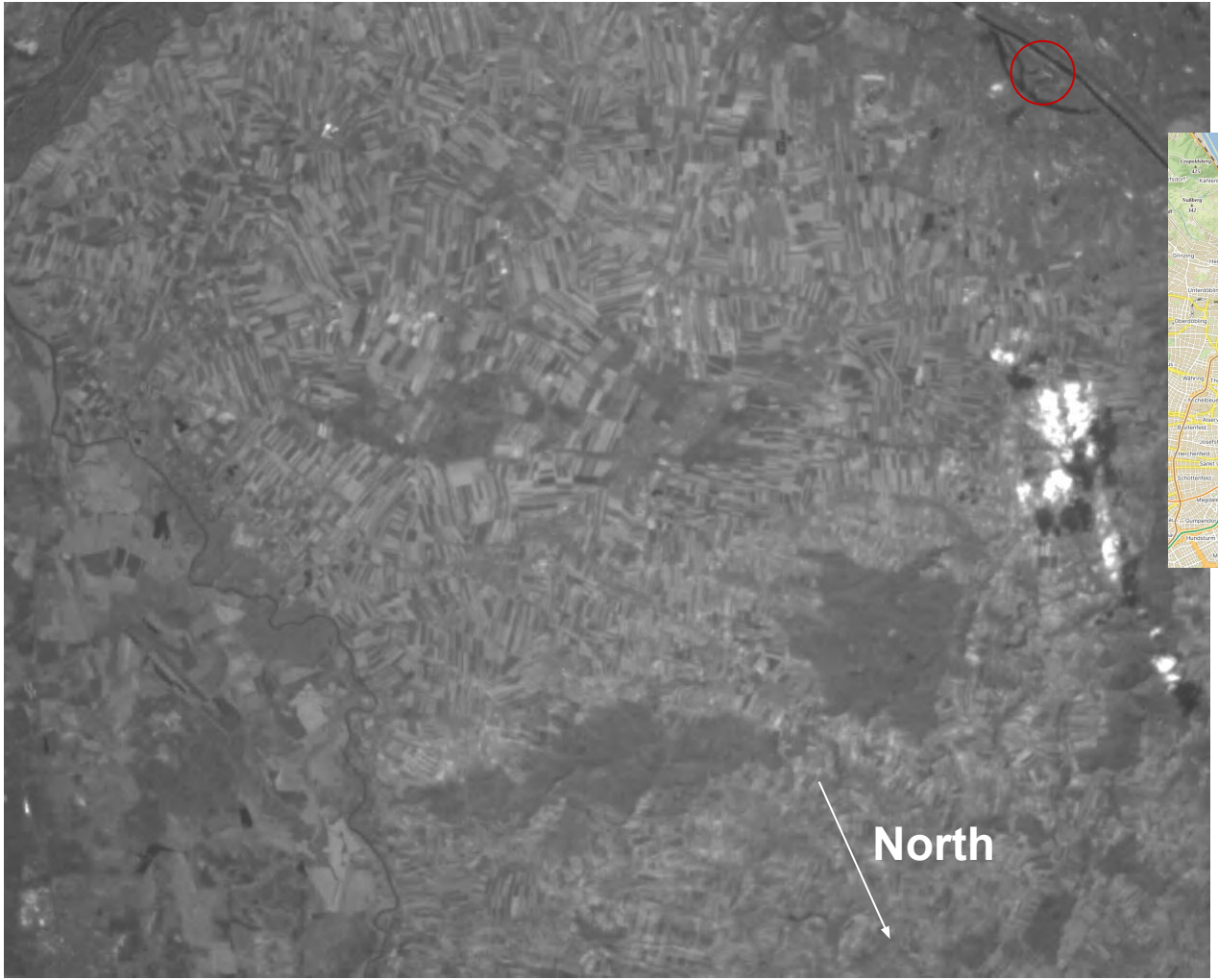
- VZLUSAT-2 (see EOSS 2023 talk)
  - It is orbiting for 2.5 years
  - Two Linux-based computers
    - Neither degraded performance nor data loss so far
  - Scientific payloads, technology demonstrations
  - Low Earth Orbit (~530 km), now <480 km
  - Size: 3U (10 x 10 x 34 cm / 4" x 4" x 13.4")
  - Mass: 4 kg / 8.8 lb



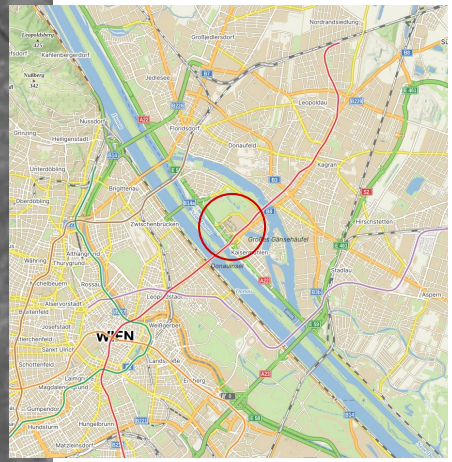




Source: VZLUSAT-2, Eltvor's instrument



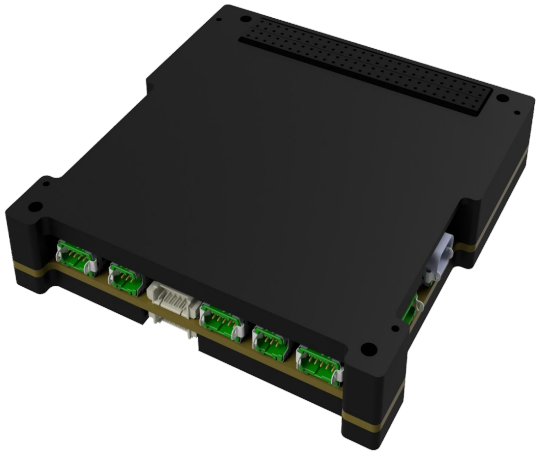
Source: VZLUSAT-2, Eltvor's instrument



Source: Mapy.cz

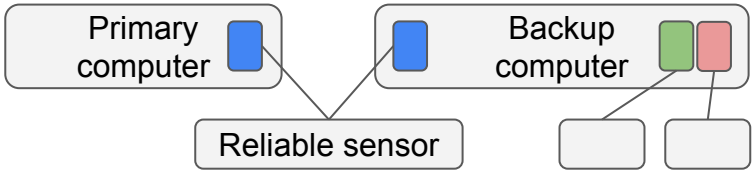
VZLÚ's approach on NewSpace 2/2

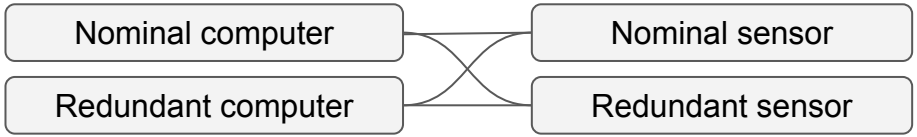
- ZHP – Zynq High-performance Platform
  - Linux-based computer based on COTS components
  - Target applications
    - Payload controller
    - Data handling subsystem
    - Attitude determination and control system
  - Tests
    - Thermal-vacuum cycling (see later)
    - Vibrations
    - Functional
    - Radiation on the component level
  - Upcoming missions: VZLUSAT-3A and VZLUSAT-3B, CORVUS, stratospheric flight

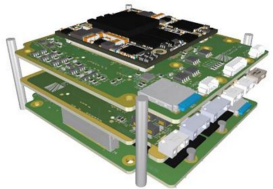


Fault mitigation on a satellite

- Failure analysis (e.g., FMEA)
- Avoid single point of failure by redundancy
  - Redundant functionality
    - The backup computer has its primary function (green and red rectangles). It may have the functionality (blue rectangle) of the primary computer.
  - Cross strapping
    - Similar to above, but more resource-intensive
  - Dual boards
    - See Fraunhofer EMI presentation at EDHPC 2023
- Don’t run it if not necessary





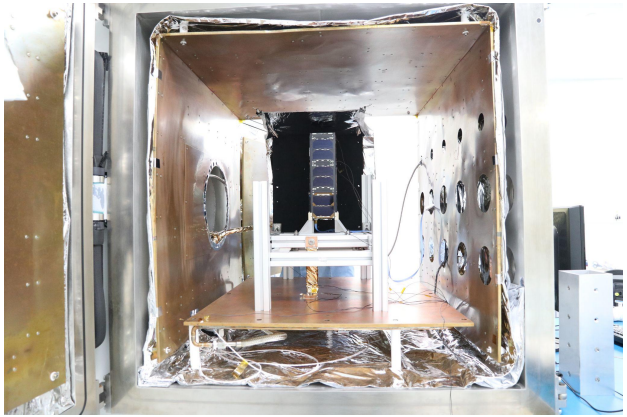


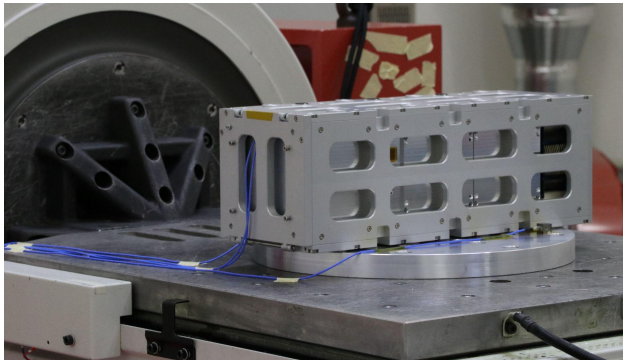
Source: Fraunhofer EMI presentation

Fault mitigation on the subsystem (payload) level

See our talk at EOSS 2023. In brief:

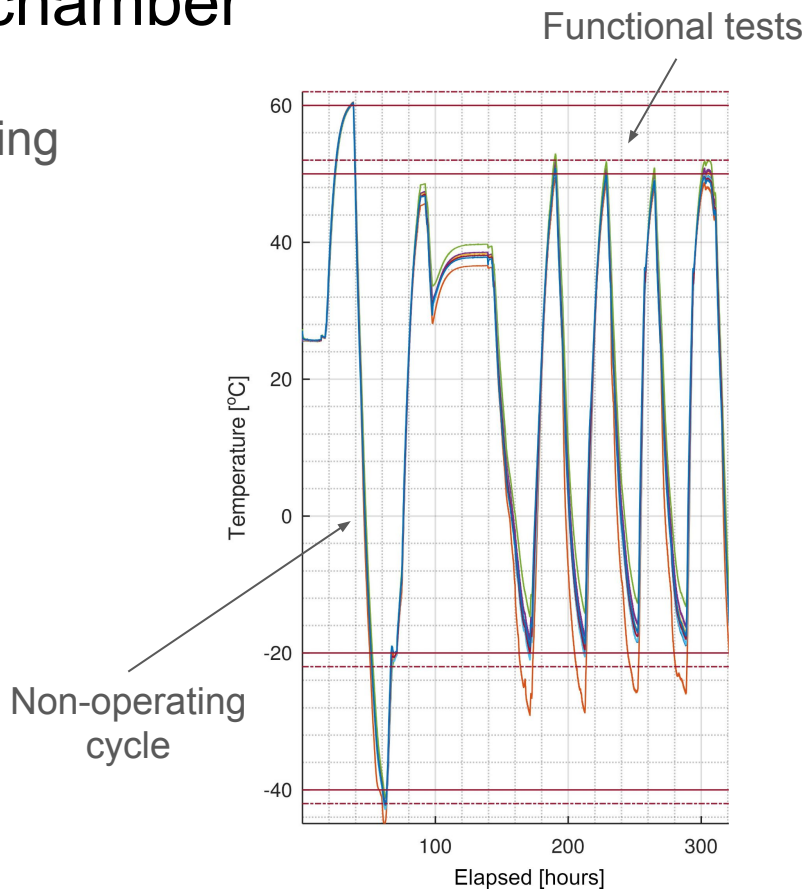
- Selection and testing of COTS components
- Hardware design
- Production (e.g., PCB certifications)
- Testing
  - Thermal
  - Mechanical (shock, vibration)
  - Radiation testing (Total Ionizing Dose and Single Event Effects)
  - Functional (short and long-term, day in the life)
  - Early integration with close subsystems (e.g., sensor and its controller)
  - System/subsystem/component level





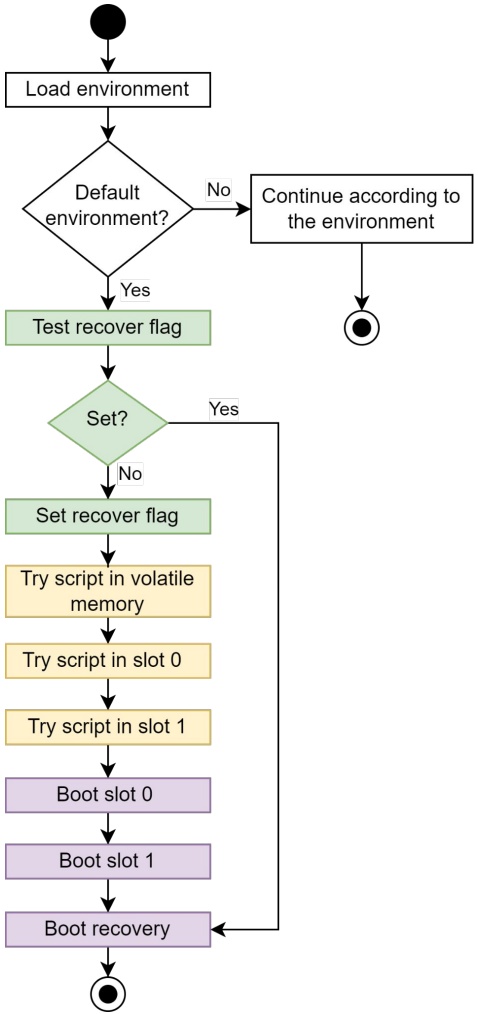
Testing in the thermal-vacuum chamber

- Temperature reference points on the testing structure (not the CPU/memory)
- Pressure ~10^{-3} Pa
- 60 °C = 140 °F
  50 °C = 122 °F
  -20 °C = -4 °F
  -40 °C = -40 °F
- We have in-house thermal-vacuum chamber



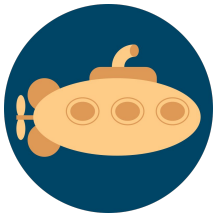
Our payload computer design principles

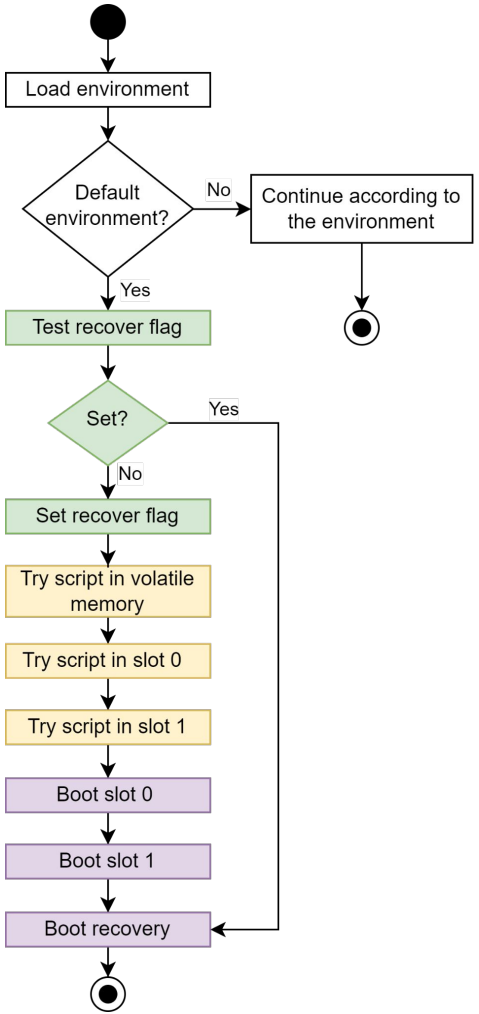
- Autonomous, but dumb
  - Try hard to boot to Linux and then wait for a command (from either satellite on-board computer or operator)
  - Simplified U-Boot flow →
  - Examples
    - Iterate through boot images
    - Don’t start the services automatically
    - Neither power-on nor mount an external storage
- Minimize writes to non-volatile memories
- Duplicate (at least) critical system images (e.g., bootloader, kernel, rootfs)



Configuration of Das U-Boot and Linux 1/2

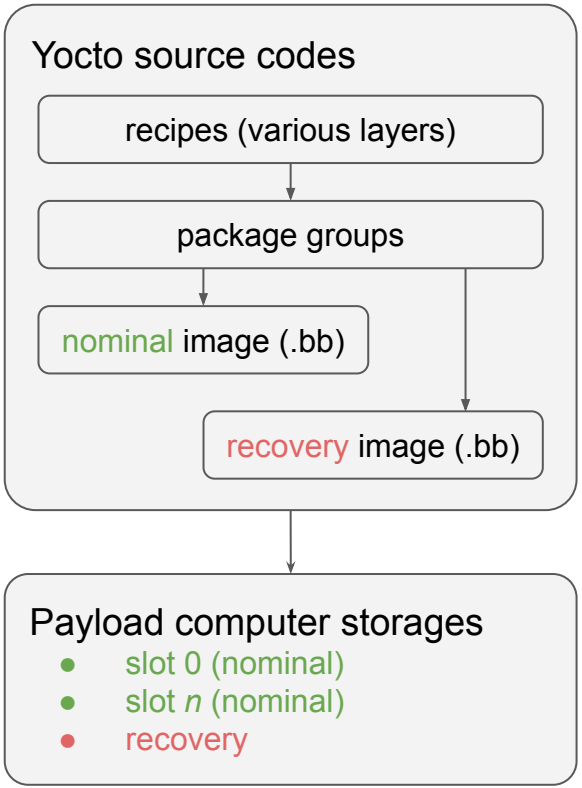
- No changes in the source code, only configuration
  - Rely on what is used by most users
- Build is handled by Yocto
  - Avoid post-build modifications because of traceability
- Das U-Boot
  - The image is duplicated
  - The image is protected by checksum
  - It handles boot to the recovery system
  - Boot image selection implemented in the boot script





Configuration of Das U-Boot and Linux 2/2

- Linux and root filesystem
  - Nominal and recovery rootfs are Yocto images based on the shared recipes in layers.
  - Nominal Linux image and rootfs
    - Reboot on any detected error (e.g., reboot on panic)
    - Careful hardware and software watchdog settings
  - Recovery Linux image and rootfs
    - Different configuration and build
    - It tries to run despite errors detected
    - System analysis and recovery by operator
    - Make sure you know you are in the recovery mode



Conclusion/Wiki

- Where do you find this presentation?
- Do you want to share your knowledge with space app?
- Are you interested in space app?